Uvod v podatkovne baze

Motivacija – študij primera

- Od naročnika 'Virtualna šola Miki Miška' smo dobili nalogo narediti sistem, ki bo omogočal hranjenje podatkov o študentih, tečajih, profesorjih, voditeljih in udeležencih posameznih tečajev. Aplikacija naj omogoča:
  1. Hranjenje podatkov za daljše časovno obdobje ➔ lahko sklepamo, da se bo s časom nabrala velika količina podatkov (tudi več 100MB)
  2. Zaščito pred nesrečami in zaščito pred nepooblaščeno uporabo podatkov.
  3. Izvajanje različnih poizvedb (primer: Kateri učitelj poučuje predmet MAT4).
  4. Dodajanje, spreminjanje in brisanje podatkov.
  5. Sočasen dostop do podatkov več 10 oz. 100 uporabnikom.
  6. Skrbnik sistema mora imeti možnost za dodajanje in spreminjanja tipov podatkov, uporabnikov aplikacije, ….

1. premislek

- Za programiranje izberemo C++
- Podatke bomo shranjevali v datoteke
- Datotečno strukturo bomo zapisali v kodo programa

- Problemi:
  1. Nadzor sočasnosti dostopov
  2. Izbira algoritmov
  3. Veliko različnih poizvedb
  4. Zagotavljanje nedeljivosti transakcij
  5. Izvoz podatkov v druge formate
  6. Spremembe strukture podatkov

- Rešitev:
  1. Vse bomo sprogramirali
  2. V rezervi bomo vedno imeli vsaj 2 programerja, ki bodo pisali nove funkcije, za izvedbo različnih poizvedb nad podatki.
  3. Za letno vzdrževanje aplikacije bomo od uporabnikov zahtevali vsaj toliko denarja, koliko stane prvotni razvoj aplikacije.

- **Težava: Uporabniki niso pripravljeni plačati tako visoke cene vzdrževanja!**

2. premislek

- Slišali smo za nek nov način shranjevanja podatkov v podatkovne baze, pri katerem večina prejšnjih težav odpade ali se bistveno zmanjša.
- Uporabnikom predlagamo drugačen način implementacije sistema z bistveno nižjo ceno vzdrževanja.

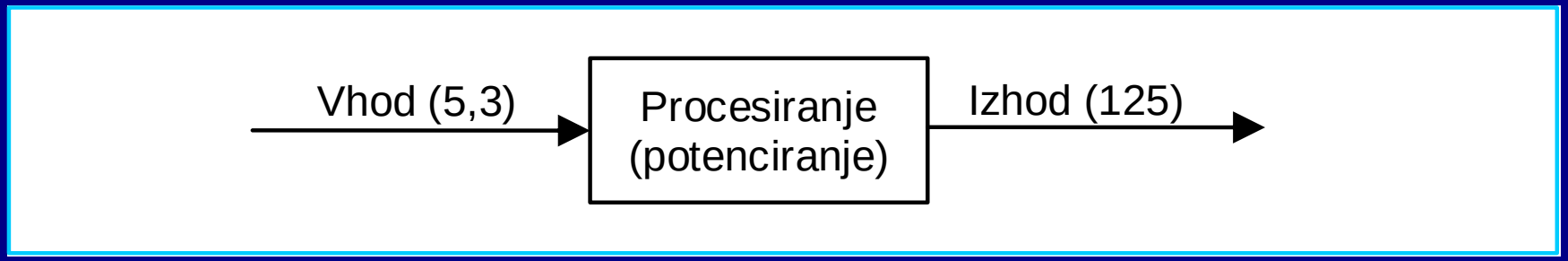
- Uporabniki se strinjajo!

- Dobro, dobili smo posel in se moramo hitro naučiti delati s podatkovnimi bazami.

PB - Trajno hranjenje podatkov?

1. primer

Danes			Čez 14 dni	
Koliko je 5^3?	Odgovor = 125		Koliko je 5^3?	Odgovor = 125

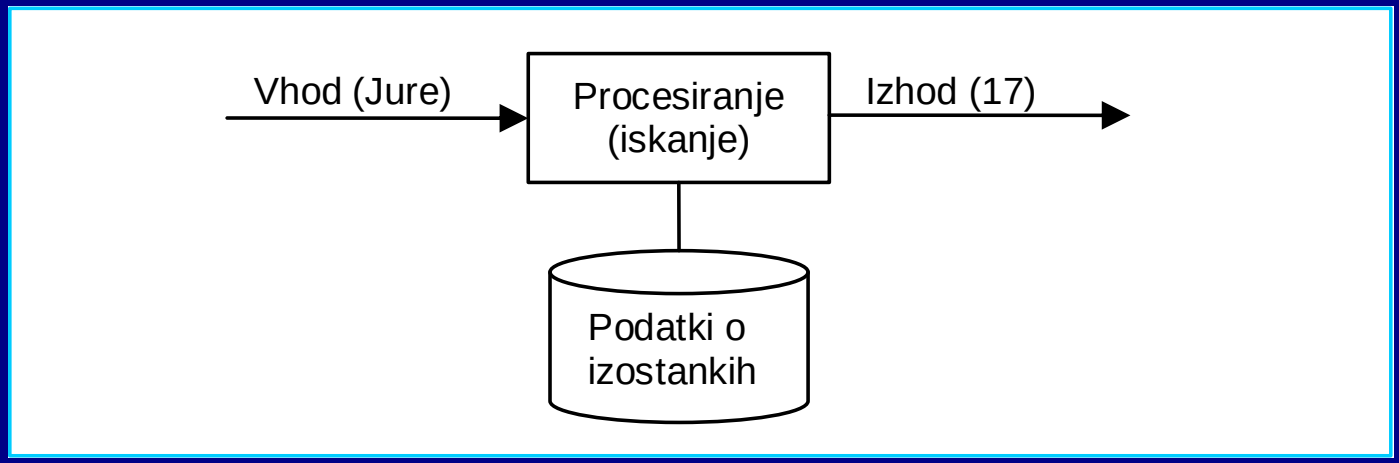


Trajno hranjenje podatkov v tem primeru ni potrebno!

PB - Trajno hranjenje podatkov (nad.)

2. primer

Danes			Čez 14 dni	
Koliko (ne)opravičenih izostankov ima moj sin Jure?	Odgovor = 2		Koliko (ne)opravičenih izostankov ima moj sin Jure?	Odgovor = 17 (ups!!!!)

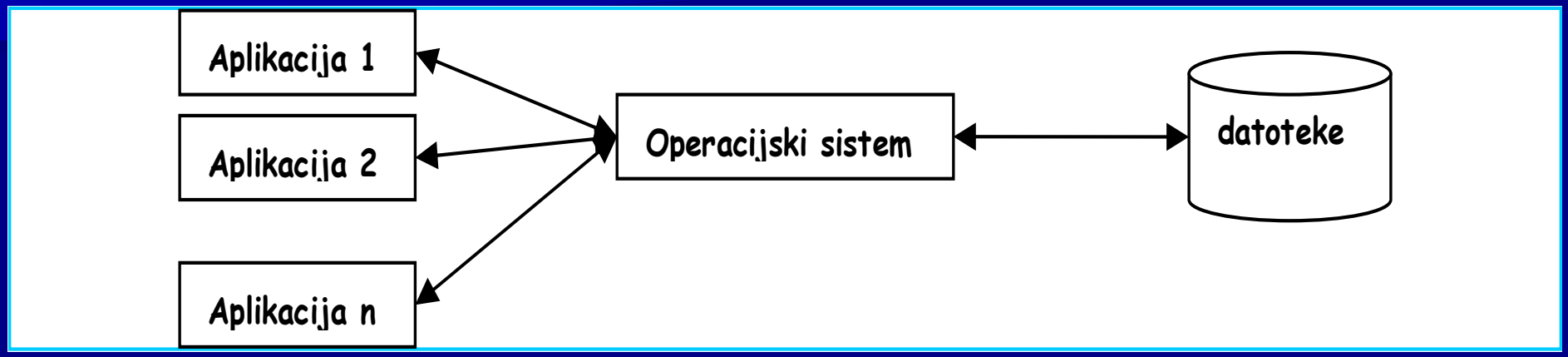


Trajno hranjenje informacije / podatka oziroma trajna informacija = informacija, katere življenjski čas je daljši od življenjskega časa enega procesa (poenostavljeno gledano izvedbe programa)

Načini implementacije trajnosti podatkov

- ‘klasični’ (datoteke)
- uporaba PB in SUPB

Klasični načini implementacije trajnosti

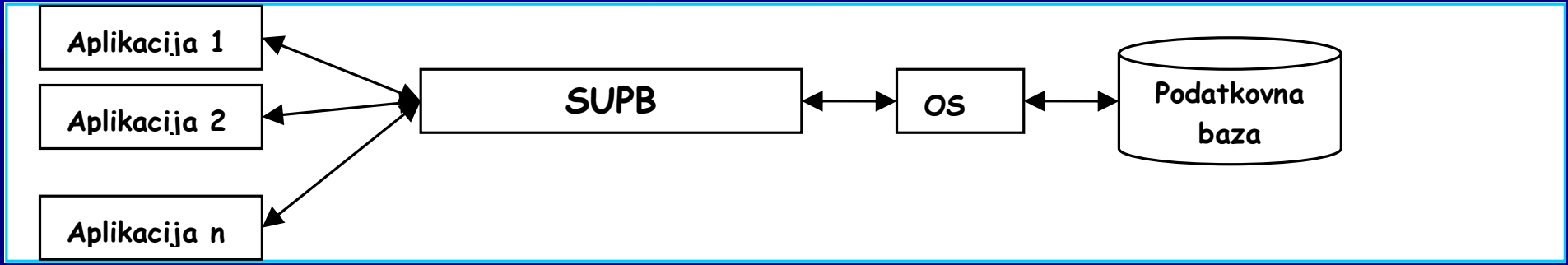


- Datoteka na nivoju OS = zaporedje zlogov
- Programi za obdelavo podatkov morajo poznati strukturo podatkov - zapisana je v kodi programa
- OS ne more preprečiti napak, ker ne pozna strukture datotek.
- Vse zaščite in druge omejitve morajo biti realizirane znotraj programa!
- Če več programov oziroma uporabnikov sočasno uporablja eno datoteko -> programsko zagotavljanje zaščite podatkov (v vseh programih)

Slabosti ‘klasičnih’ sistemov

- ☹ časovno požrešna metoda razvoja programske opreme (veliko programiranja)
- ☹ ni podpore 'ad hoc' poizvedbam
- ☹ izolirani ‘otoki’ informacij (ločene datoteke po različnih oddelkih podjetja)
- ☹ močna odvisnost med podatki in programi (sprememba strukture datoteke zahteva tudi spremembo vseh programov, ki jo uporabljajo)
- ☹ večkratno zapisovanje istih podatkov ➔ nenadzorovana redundanca podatkov ➔pri dodajanju, spreminjanju in brisanju podatkov pogosto prihaja do anomalij med podatki
- ☹ Pomanjkanje kontrole ➔ nekonsistenca/neskladnost podatkov
- ☹ Podatki, shranjeni s pomočjo ene aplikacije ne morejo biti obdelani s strani druge aplikacije (nekompatibilnosti formatov)

Uporaba PB in SUPB-ja za implementacijo trajnosti



- struktura shranjenih podatkov mora biti podana na način, ki ga SUPB razume; primer:

```
Create Table "Dijak"  (
        "DijakID" Integer NOT NULL,
        "Priimek" Char(20) NOT NULL,
        "Ime" Char(10) NOT NULL,
        "Razred" Char(3),
        Primary Key ("DijakID")
);
```

- SUPB sam odkriva in preprečuje napake
- Programiranje se poenostavi in se prestavi na višji nivo abstrakcije.
- SUPB - vmesnik med OS in aplikacijskimi programi / uporabniki

Uporaba PB in SUPB-ja za implementacijo trajnosti (nad.)

- Koda SUPB-ja je testirana in dobro optimizirana
- SUPB-ji imajo vgrajene rutine za razvrščanje in iskanje podatkov, za upravljanje z izrabo datotečnih izravnalnikov (bufferjev), za shranjevanje datotek, statistične funkcije, …
- SUPB je optimiziran za delo z velikimi količinami podatkov, podpira večuporabniško okolje, varnostne mehanizme, ki preprečujejo dostop do podatkov nepooblaščenim osebam in mehanizme, ki zagotavljajo zaščito podatkov v primeru porušitve sistema ter obnavljanje sistema.
- PB je dostopna le s pomočjo SUPB ➔ omogočeno je skrivanje / prikrivanje internih sprememb znotraj podatkovne baze uporabnikom in uporabniškim programom. Vodilo abstraktnih podatkovnih tipov je možnost implementacije sprememb na nižjem nivoju in hkrati ohranjanje vmesnika proti višjem nivoju.